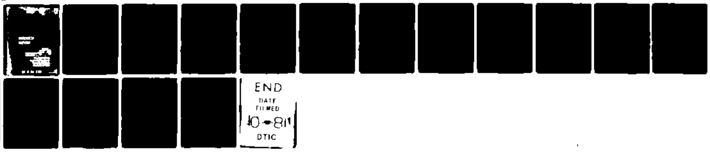


ON THE PALLET LOADING PROBLEM

Research Report No. 81-11

by

Thom J. Hodgson

August 1981

Department of Industrial and Systems Engineering
University of Florida
Gainesville, Florida 32611

This research was supported in part by the U.S. Air Force, under contract number F73AFL-00360001, and by the Office of Naval Research, under contract number N00014-76-C-0096.

[illegible]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 81-11	2. GOVT ACCESSION NO. AD-A103541	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ON THE PALLET LOADING PROBLEM		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER 81-11
7. AUTHOR(s) Thom J. Hodgson		8. CONTRACT OR GRANT NUMBER(s) F73AFL-00~~36~~0001 (Air Force) N00014-76-C-0096 ✓(Navy)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Industrial and Systems Engineering University of Florida Gainesville, Florida 32611		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - A.F. Logistics Arlington, VA Mgmt. Center Gunter AFS, Alabama		12. REPORT DATE August 1981
		13. NUMBER OF PAGES 17
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

Pallet Loading

Dynamic Programming

Heuristics

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper the two-dimensional pallet loading problem is considered: that is, the problem of loading a rectangular pallet of size "L" by "W", drawing from a set of "n" rectangular boxes. The objective is to maximize the area covered on the pallet by the boxes loaded. The problem is approached using a combination of Dynamic Programming and heuristics. The structured solutions resulting from the application of the "dynamic program" have two serendipitous characteristics: any item may be placed on the periphery of the pallet for easy access, and some control may be retained over the center of gravity of the pallet. Ways of using the procedure to load three-dimensional pallets are discussed. Computational results are given.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

ABSTRACT

ON THE PALLET LOADING PROBLEM

In this paper the two-dimensional pallet loading problem is considered: that is, the problem of loading a rectangular pallet of size "L" by "W", drawing from a set of "n" rectangular boxes. The objective is to maximize the area covered on the pallet by the boxes loaded. The problem is approached using a combination of Dynamic programming and heuristics. The structured solutions resulting from the application of the "dynamic program" have two serendipitous characteristics: any item may be placed on the periphery of the pallet for easy access, and some control may be retained over the center of gravity of the pallet. Ways of using the procedure to load three-dimensional pallets are discussed. Computational results are given.

Introduction

Much of the packaged material which is shipped in trucks, railcars, aircraft, and ships is packed on a pallet or in some other bulk container. The packing problem can be stated simply as trying to pack as many packages as possible into a container. Certainly the general packing problem would include irregularly shaped packages and containers. However, in this paper, only rectangularly shaped packages (boxes) and containers (pallets) are dealt with. There are, at least, two major problems that can be identified as "The Pallet Packing Problem." The first problem could be called "The Manufacturer's Pallet Packing Problem." In this problem, the manufacturer produces a product which is packaged in identical boxes; the boxes may be packed in identical cartons; the cartons are packed on identical pallets; and the pallets are loaded in standard sized trunks, railcars, or shipping containers. The problem is to choose the package, carton, pallet, (and possibly the container) dimensions so that the volume of product packed in a container is maximized. This problem requires a one-time analysis to find the solution. With the exception of Steudel [10], little has appeared on this problem in the open literature. However, it is clear that industry is attacking this problem and several consulting firms offer services in this area.

The second problem could be called "The Distributor's Pallet Packing Problem." In this problem, the distributor fills an order from a customer. The order is packaged in boxes of varying dimensions. The problem is to pack the boxes on a standard pallet so as to maximize the volume placed on each pallet (i.e., minimize the number of pallets used to ship the order). The problem requires a new analysis for each pallet packed. As a consequence, from an economic stand point, the cost of a solution for the Distributor's Problem can be, at most, a fraction of the cost of a solution for the Manufacture's Problem. In addition, in most applications, the Distributor's Problem must be solved quickly (i.e., real-time computation) in order for the solution to be applied.

For the Manufacturer's Problem, present technology supports the packing of pallets using automated material handling systems. However, the Distributor's Problem, by its nonrepetitive nature and solution time requirements, is more difficult. In order to automate the physical packing of a distributor's pallet, one first needs a packing algorithm which essentially is real-time.

The problem addressed in this paper is a constrained version of the Distributor's Problem. Some of the boxes to be packed on the pallet may contain volitile liquids or explosives. As a consequence, those items must be packed on the periphery of the pallet so that, if necessary, they can be removed quickly. This problem is faced by the U.S. Air Force when they transport palletized cargo consisting of military equipment and supplies.

Background

The pallet loading problem is related to a problem long studied in the Operations Research literature: The Cutting Stock Problem. The cutting stock literature is not discussed here, but the interested reader is directed to a recent review paper by Golden [5]. The two-dimensional cutting problem (of which the pallet loading problem is a special case) has been studied by Christofides and Whitlock [1], Gilmore and Gomory [4], Hahn [6], and Herz [7]. The two-dimensional cutting problem becomes the two-dimensional pallet loading problem when the requirement for guillotine type cuts is dropped (i.e.,straight cuts made in stages from one edge to the opposite edge of the object being cut, such as with a common paper cutter). Steudel [10] studied the pallet loading problem. However, his work was limited to the case where all boxes have the same length and width. He developed a procedure combining heuristics and dynamic programming.

DeSha [2], in an unpublished master's thesis, developed a heuristic for loading containers. His procedure first sets up stacks of items to fit the container height, then loads the stacks in the container to maximize the container floor area covered. The heuristic appears to obtain quite good results using a data base with boxes whose dimensions are randomly generated. The procedure does not, however, include considerations of center of gravity, positioning of hazardous material in the container, or box manipulation (no "this end up" assumption).

The pallet loading problem falls in the category of problems called NP-HARD [3]. Consequently, a truly efficient optimal algorithm is not likely to be forthcoming. In developing an approach to the problem that would be consistent with the special needs of the USAF, it also became clear that it would be highly desirable for the system to be interactive. This would allow a user to guide the solution of a particular loading problem in order to deal with those unquantifiable elements of a "real world" loading problem. With these observations in mind, a system called IPLS (Interactive Pallet Loading System) was developed. A partial description of that system is in [8] (IPLS is undergoing evaluation and further development by the USAF Logistics Management Center). In this Paper, the algorithmic developments and conceptual use of such a system is discussed.

In the following, a pallet loading procedure is developed for the two-dimensional loading problem. Then ways of using this procedure to load real-life (three-dimensional) pallets is discussed. Computational experience is presented.

Pallet Loading Procedure

The pallet loading procedure can be described as a combination of the principles of Dynamic Programming [9] and heuristics. To understand the procedure, it will be useful first to consider a "best" procedure. It will be obvious that the "best" procedure is computationally infeasible. Therefore, structural limitations will be introduced which limit the computational effort. A serendipitous by-product of the resulting procedure structure is that positioning of hazardous material and considerations of center of gravity can be handled without loss within the procedure.

Assume that it is desirable (no matter what the cost) to find solutions to the two-dimensional pallet loading problem which maximize the area covered on the pallet. In order to achieve this end, let us consider Dynamic Programming as a solution methodology. The following definitions will be useful.

P	= A partition dividing the pallet into two parts (see figure 1). The left-hand sub-pallet must include the origin (0,0), and the right-hand sub-pallet must include the point (L,W).
i	= The index of boxes to be loaded, i=1,...,n.
l(i)	= The length of box i.
w(i)	= The width of box i.
S(i)	= The profile (shadow) of box i. The profile is a rectangle with length l(i) and width w(i).
N	= Set of all boxes to be considered for loading (of size n).
I	= Subset of the boxes, 1,2,...,n.
f(P,I)	= The maximum area which can be covered of the left-hand sub-pallet of P using the subset of boxes I.

The Dynamic Programming equation for the pallet loading problem can be given as follows:

$$(1) \quad f(P,I) = \max_{i \in I} [l(i)*w(i) + f(P-S(i),I-i)].$$

It should be noted that the notation 'P-S(i)' represents a partition which, in a graphical sense, is the partition P with a profile of box i removed from the right-hand edge of the left-hand sub-pallet. Typically, there could be many different realizations of 'P-S(i)' that should be considered within a dynamic optimization. There are other obvious difficulties with implementing equation (1). The most obvious is that the number of possible partitions P of the pallet is extremely large. This means that the state space for the Dynamic Program will require large amounts of computer storage. It also means that the computer time required to solve the Dynamic Program likely would be well beyond any sensible limit for real world applications.

One approach to developing a more tractable procedure is to limit in some way the form of the possible partitions of the pallet. In the present case, partitions of the pallet have been limited to rectangles (figure 2a). In order to specify the Dynamic Program resulting from the rectangular partitions, the following additional definitions are needed.

x,y	= Two-dimensional index specifying a rectangular partition (figure 2a).
J	= Subset of the boxes, 1,2,...,n.
g(x,y,I)	= The maximum area which can be covered of the left-hand sub-pallet of x,y using the subset of boxes I.
h(x,y,x',y',I)	= The maximum area which can be covered of the left-hand sub-pallet of x',y' less the left-hand sub-pallet of x,y using boxes from the set I (not all elements of I necessarily are used, see figure 2b).

The Dynamic Programming equation for the pallet loading problem (limited to rectangular partitions) can be given as follows:

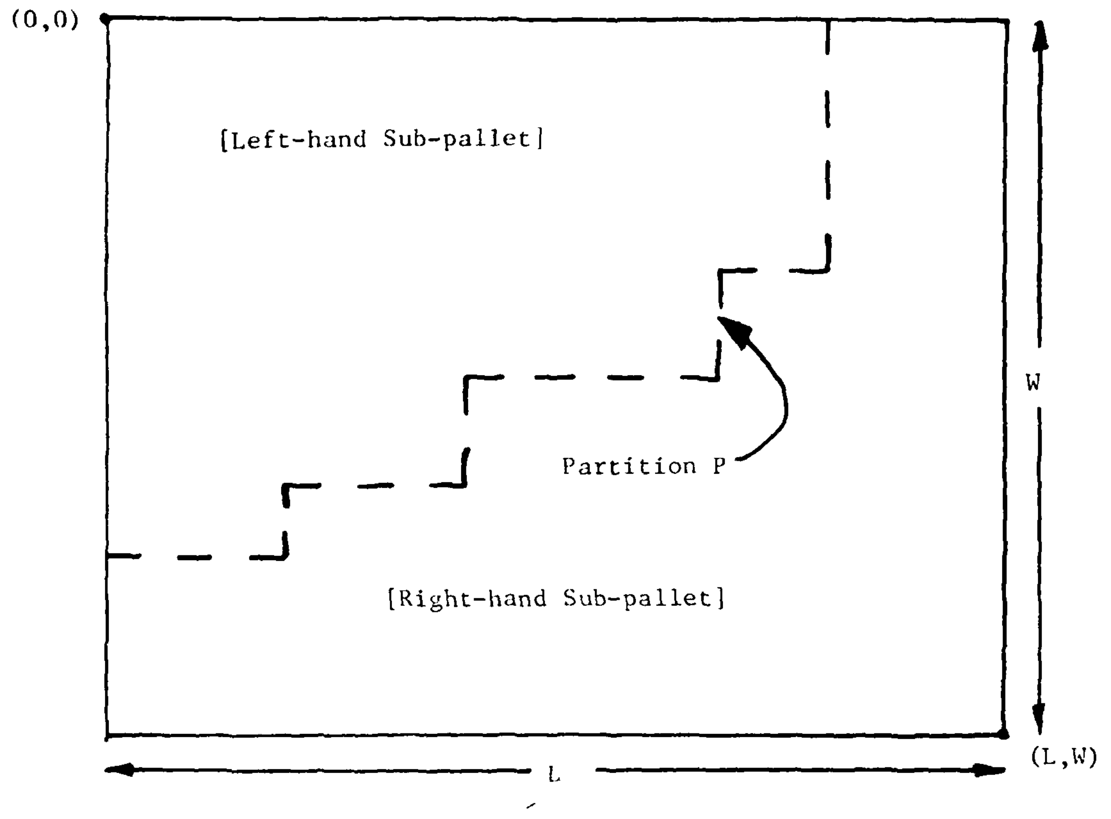


Figure 1

Plan View of Pallet With
Sample Partition P

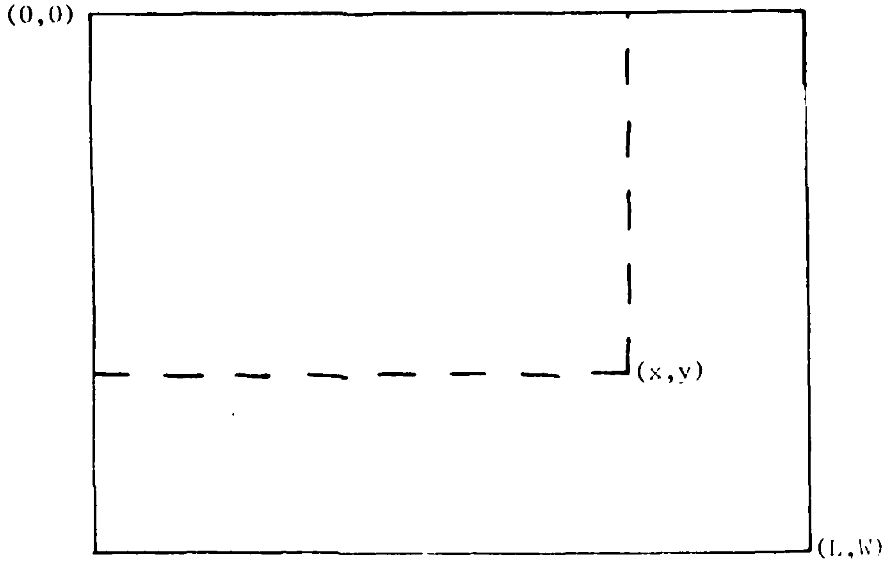


Figure 2a
Pallet With Rectangular Partition (x,y)

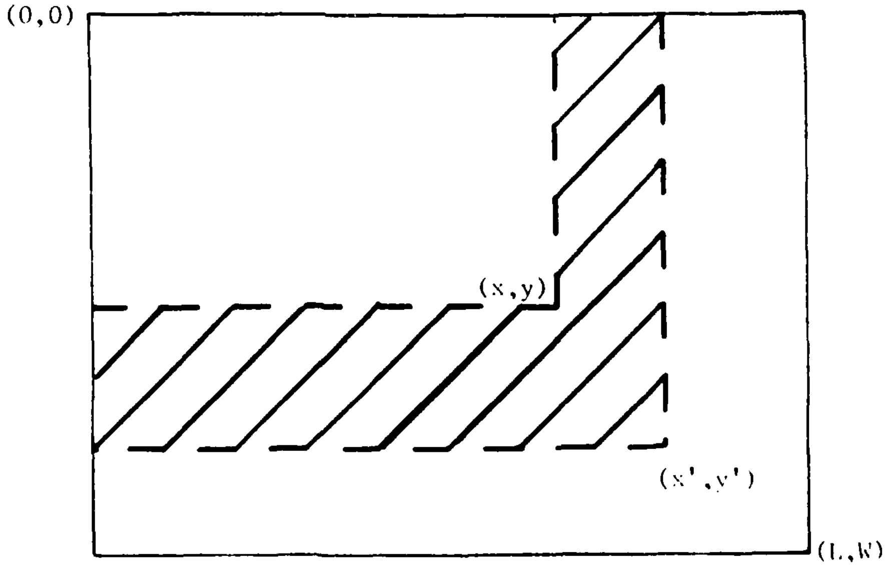


Figure 2b
Pallet With Two Rectangular Partitions
(x,y) and (x',y')

$$(2) \quad g(x',y',I) = \max_{\substack{x<=x' \\ y<=y' \\ J \subset I}} [g(x,y,J) + h(x,y,x',y',I-J)]$$

The implementation of equation (2) also has its difficulties. The function h(x,y,x',y',I-J) itself requires an optimization in order to pack the L-shaped area common to the partition x',y', but not common to the partition x,y (i.e., I-J is the cross-hatched area in figure 2b). The state-space is still too large to deal with on a practical basis.

In order to limit the size of the state space, it is necessary to carry only one partial solution (g(x,y,I)) for each partition x,y. The obvious choice is to carry

$$\max_{I} [g(x,y,I)].$$

With this limitation on the state space, the implementation of equation (2) is relatively straightforward. It is necessary, however, to specify several important details first:

1. an optimization structure for h(x,y,x',y',I-J);
2. bounding rules for the elimination of partial solutions;
3. bounding rules for the minimization of computational effort.

The optimization for h(x,y,x',y',I) is done simply by breaking the L-shaped area into two rectangular areas (figure 3) and filling each area using a linear Dynamic Programming procedure. The following definition is useful.

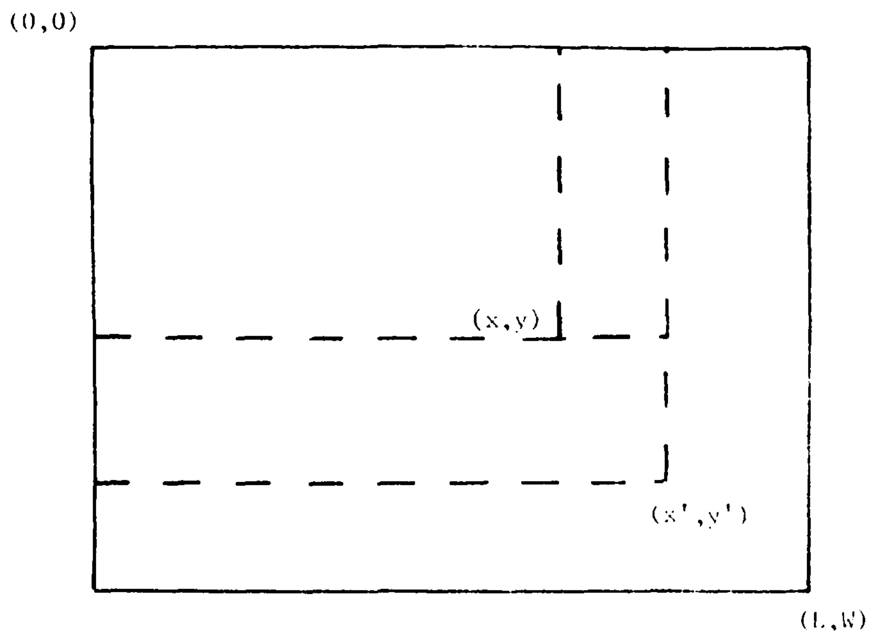
b(J,x) = The maximum possible space covered in a rectangular area of length x by stacking boxes from the set 1,...,J (Note the limitation of 'linear' stacking imposed).

The Dynamic Programming equation for the rectangular loading problem can be given as follows:

$$(3) \quad b(J,x) = \max [b(J-1,x),\ b(J-1,x-l(J))+l(J)*w(J)].$$

The implementation of equation (3) is achieved by first eliminating boxes too large to fit in the rectangle, then turning each box in the candidate set so that its longest dimension is perpendicular to the long dimension of the rectangular area (if the longest dimension of the box is less than or equal to the short dimension of the rectangular area, that is). This insures an optimal packing of the rectangular area. The L-shaped area is broken into two rectangular areas (corridors) two different ways (figure 3) for the application of equation (3). The best solution obtained, in terms of area covered, is retained.

A simple, and almost obvious, bounding rule that eliminates a great deal of the storage requirements for the state space in computing equation (2) is that a partial solution does not need to be retained for the partition x,y if there exists a partial solution for a partition x',y', (x'<=x, y'<=y) such that the



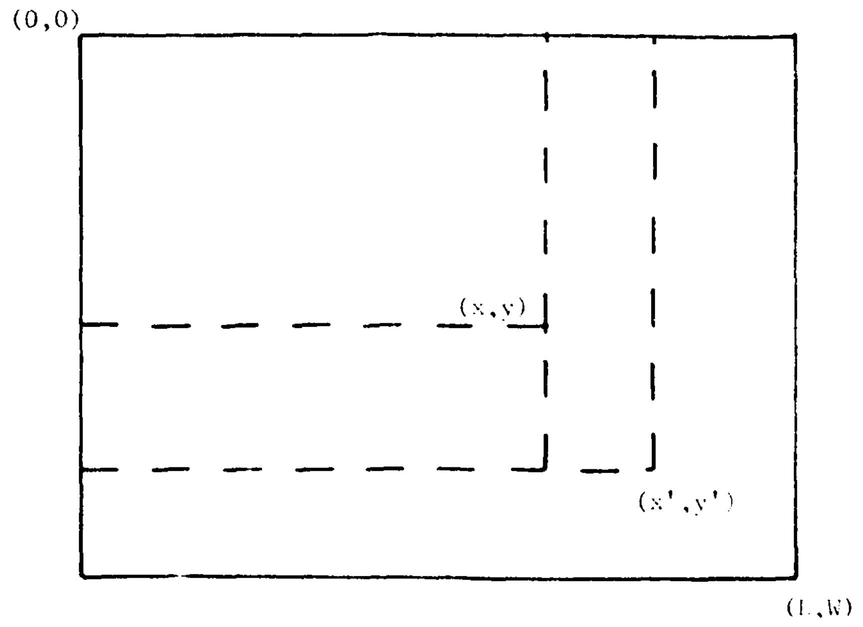


Figure 3
Two Ways To Break Up L-Shaped Area

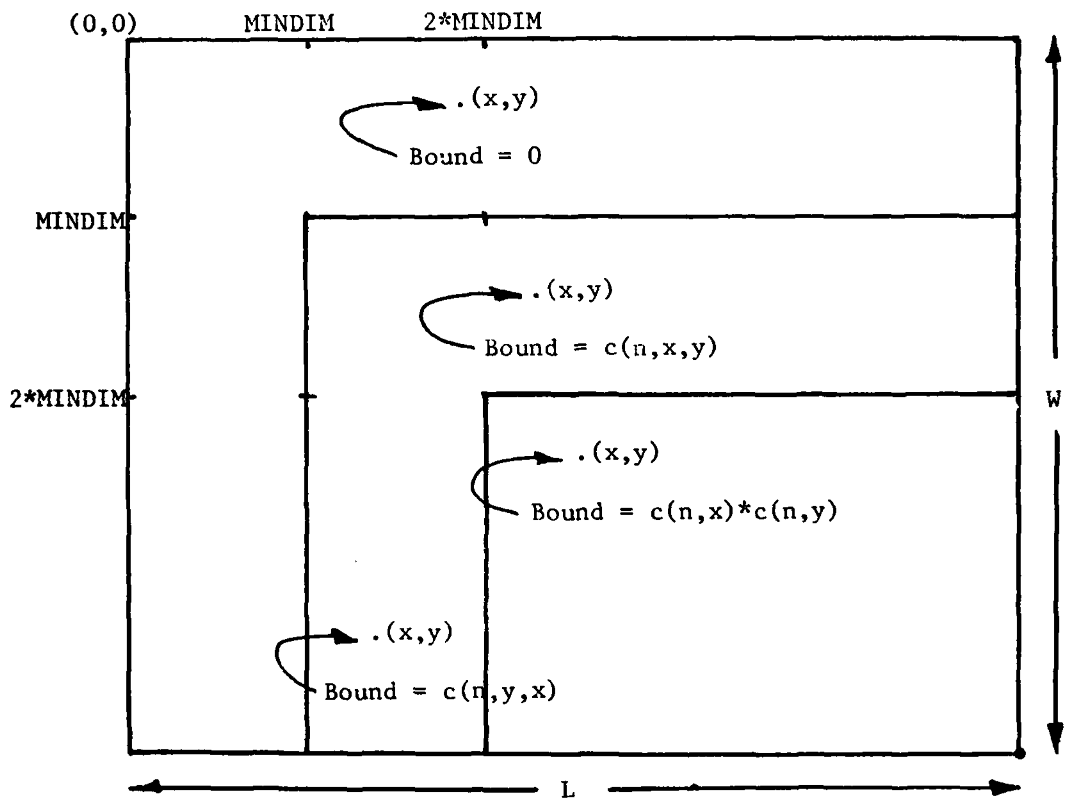


Figure 4

Matrix Layout of Bounds for
Pallet (sub-pallet)

solution value for x',y' (max [g(x',y',I)]) is greater than or equal to the solution value for x,y (max[g(x,y,I)]).

Another effective bounding scheme is used to eliminate computation time. It involves computing an upper bound on the amount to be loaded in a rectangular area of dimension LxW by using the result of a linear Dynamic Program in multiplicative fashion. The following definition is useful.

c(j,x) = The maximum possible linear space covered in a length x by stacking boxes from the set 1,...,j.

The Dynamic Programming equation for the linear loading problem can be given as follows:

(4) c(j,x) = max[c(j-1,x), c(j-1,x-l(j))+l(j), c(j-1,x-w(j))+w(j)]

The function c(n,x) specifies the maximum linear coverage that is possible on the line segment [0,x] choosing from the set of boxes 1,...,n (positioning them by either length or width). For a pallet (rectangular sub-pallet) of size L by W, an upper bound on the maximum load (coverage) possible is c(n,L)*c(n,W). The function c(n,x) can be computed prior to the pallet loading and is easily implemented within the structure of equation 2. The upper bound can be used to eliminate the need to consider a given partial solution (x',y') in equation (2) altogether. It can also be used to eliminate the computation of Dynamic Program equation (3) within the optimation of equation (2) when considering a specific partial solution (x,y).

A more powerful bounding procedure can be used for rectangular sub-pallets of certain dimensions. Clearly, if the width of the sub-pallet is less than the minimum dimension (length and/or width) in the candidate box set, it is impossible to pack any of the candidate boxes on the sub-pallet. Consequently, the upper bound on the amount which can be loaded is zero. Now, let MINDIM equal the minimum dimension in the candidate box set. If the width of the sub-pallet satisfies

MINDIM <= W < 2*MINDIM,

then boxes not fitting within the width of the sub-pallet can be eliminated from the computation of equation (4), and the upper bound function is just c(n,x,W) (where the "W" indicates the elimination of non-fitting box lengths and/or widths from the candidate set, i.e., l(j)>W and/or w(j)>W). The matrix layout of the bound is shown graphically in figure 4.

Another bound to suppliment the above bounds can be calculated. The following definitions are useful.

a(i) = The area of box i (i.e., a(i) = l(i)*w(i))

d(j,z) = The maximum possible area covered on a pallet of area z by loading boxes from the set 1,...,j, and ignoring considerations of box shape.

The Dynamic Programming equation for the area loading problem can be given as follows:

(5) $d(j,z) = \max [d(j-1,z), a(j)+d(j-1,z-a(j))]$

The function d(j,z) specifies the maximum area coverage that is possible on a pallet of size z, choosing from the set of boxes 1,...,n, and assuming that the boxes can be "mashed" into any shape without loss of area. The upper bound for a rectangular pallet (sub-pallet) of size W by W, is just d(n,L*W). The upper bound used in the procedure, then, is just the minimum of all the bounds described above.

The solution procedure results in the boxes being placed in corridors on the pallet. Two serendipitous by-products occur. First, since each corridor has at least one end on the pallet perimeter, any box which contains hazardous material can be placed at the end of the corridor (as per USAF Regulations) within the structure of the solution. It is possible to get multiple "hazardous" boxes on a corridor, so there is no guarantee that any box can be placed on the periphery of the pallet. However, it has been our experience that the probability of not being able to do so is quite low. Second, also since boxes can be moved within their assigned corridors, some control can be maintained over the center of gravity of the pallet within the structure of the solution.

Since real-life pallet loading problems typically have considerably more complexity than the present formulation, the approach described above is best used as part of a highly interactive computer system [8]. The following illustrates two potential ways of using the procedure to load a three-dimensional pallet. The first approach is to load the pallet in layers. That is, from the set of boxes to be loaded on the pallet, choose a subset all of which have the same height (It may be necessary to "rotate" some boxes in order to get the correct dimension as the "height".). Then run the procedure to load the pallet using the constant height subset of boxes as input data. The tops of the loaded boxes now form a new "pallet" on which another set of boxes (of common height) can be loaded. This process can be repeated until either the total weight limit or the total height limit has been reached. The result is a pallet with a "layered" load as in figure 5a.

The second approach is to load the pallet with columns of boxes. That is, make up stacks (or columns) of boxes such that the stacks are no higher than the maximum allowable pallet height and the volume filled over the base box (bottom box in the stack) is maximized. Stacks can be made up using a simple Dynammic Programming (knapsack) procedure [9]. Using the stacks as big "boxes", the pallet loading procedure can then be used. The result is a pallet with a "stacked" load as in figure 5b.

Computations

The pallet loading procedure was programmed in Fortran IV and implimented on both a PDP-11/34 and the University of Florida Amdahl 470. The computation times shown in Table 1 are for the Amdahl using the IBM Fortran G compiler. Since the Amdahl operates under a "virtual" operating system, the reported times include "paging", and actual CPU times may be as much as

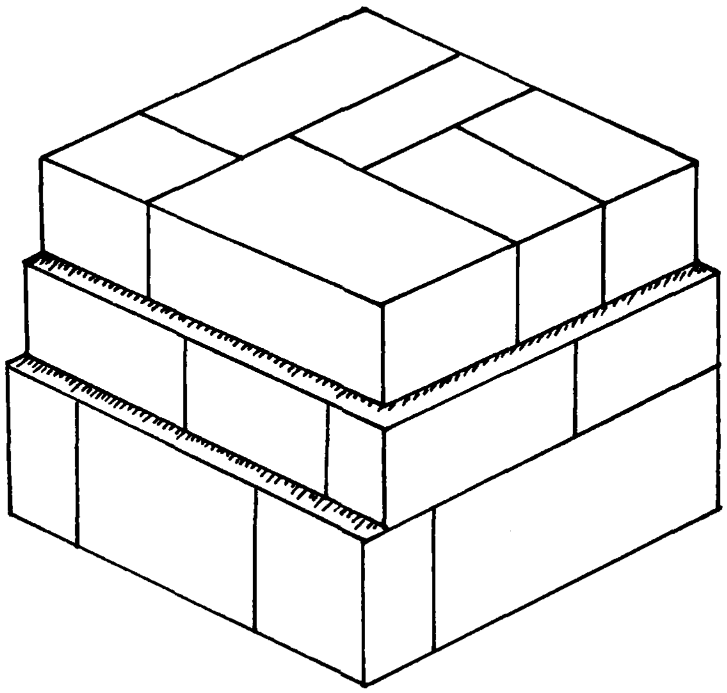


Figure 5a
"Layered" Pallet Load

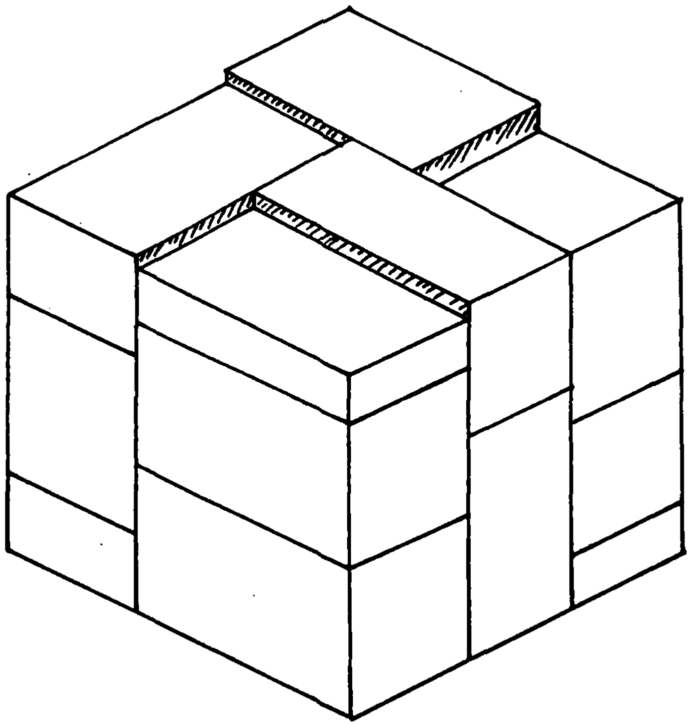


Figure 5b
"Stacked" Pallet Load

50% less (CPU times for the PDP-11/34 using the Fortran IV-Plus compiler are approximately ten times thse reported for the Amdahl.). The experimental computation runs were made using five different sets of boxes and four different pallet sizes. As might be expected, computation times grow as a function of pallet size (area) and diversity of the box dimensions in the data set (maximum box length - minimum box width). It is our opinion that further improvements in the bounding function would greatly improve observed running times. Comparison runs with and without the existing bounding function resulted in reductions of up to thirty to one in computation time. Yet it is our observation that the bounding function may still have much room for improvement.

The reported percent coverage of the pallets appears to be quite good. Average coverage (actual coverage in square inches over the upperbound in square inches, times 100) was 94.3% for all problems.

Table 1
=======

	Box Set		Pallet		Virtual	
	Minimum	Maximum			Computation	
#Boxes	Length	Length	Length	Width	Units (sec)	% Coverage
29	17	19	70	50	.09	100
29	17	19	80	60	.28	95
29	17	19	90	70	.87	89
29	17	19	104	84 *	3.04	93
25	17	21	70	50	.32	100
25	17	21	80	60	2.01	89
25	17	21	90	70	3.13	93
25	17	21	104	84 *	15.44	90
29	17	30	70	50	.36	99
29	17	30	80	60	1.61	98
29	17	30	90	70	6.49	99
29	17	30	104	84 *	**	**
25	17	36	70	50	.58	95
25	17	36	80	60	6.83	88
25	17	36	90	70	22.89	94
25	17	36	104	84 *	**	**
25	10	36	70	50	8.20	92
25	10	36	80	60	**	**

* The standard U.S. Air Force Pallet is 104 in. by 84 in.
** The problem was not solved in 25 'virtual' seconds.
,

References

[1] Christofides, N., and Whitlock, C., "An Algorithm for Two-Dimensional Cutting Problems," OPERATIONS RESEARCH, Vol. 25, No. 1, Jan. 1977, pp. 30-44.

[2] DeSha, E.L., "Area Efficient and Volume Efficient Algorithms for Loading Cargo," Masters Thesis, United States Navy Post-Graduate School, Sept. 1970.

[3] Garey, Michael, and Johnson, David, COMPUTERS AND INTRACTABILITY, W.H. Freeman, San Francisco, 1979.

[4] Gilmore, P.C., and Gomory, R.E., "Multistage Cutting Stock Problems of Two and More Dimensions," OPERATIONS RESEARCH, Vol. 13, No. 1, Jan. 1965, pp.94-120.

[5] Golden, B.L., "Approaches to the Cutting Stock Problem," AIIE TRANSACTIONS, Vol. 8, No. 2, June 1976, pp.265-272.

[6] Hahn, s., "On the Optimal Cutting of Defective Glass Sheets," IBM N.Y. Scientific Center Report No. 320-2916, 1967.

[7] Herz, J.C., "A Recursive Computing Procedure For Two-Dimensional Stock Cutting," IBM JOURNAL OF RESEARCH AND DEVELOPMENT, Vol. 16, 1972, pp. 462-469.

[8] Hodgson, Thom J., "IPLS: Interactive Pallet Loading System," Research Report No. 81-9, Industrial and Systems Engineering Department, University of Florida, Gainesville, Florida 32611, June, 1981.

[9] Nemhauser, George L., INTRODUCTION TO DYNAMIC PROGRAMMING, John Wiley and Sons, Inc., New York, NY, 1966.

[10]Steudel, H.J., "Generating Pallet Loading Patterns: A Special Case of the Two-Dimensional Cutting Stock Problem," MANAGEMENT SCIENCE, Vol. 25, No. 10, Oct., 1979, pp. 997-1004.